F1 Regularization in Computer Vision

Abhay Venkatesh
University of Wisconsin-Madison
Madison, WI-53706
abhayvenkate@wisc.edu

Hansi Zeng
University of Wisconsin-Madison
Madison, WI-53706
hzeng27@wisc.edu

Abstract

We present a method for ”F1 Regularization” with applications in Computer Vision. Inspired by recent work from [1] that allows scalable optimization of the F_1 score, we present a novel idea of using this F_1 score as a regularizer for any arbitrary computer vision task such as image segmentation or classification. Furthermore, we introduce a stochastic gradient descent algorithm for optimizing this F_1 score alongside the model parameters for the primary task. We provide many motivating applications ranging from classification, to PU segmentation, and machine learning fairness. Finally, we evaluate our model on segmentation and classification, and beat the baselines by a significant amount in both cases.

1. Introduction

1.1. Motivation

In your quintessential supervised machine learning problem, we learn $h_\Theta(\mathbf{X})$ by solving

$$h_\Theta(\mathbf{X}) = \arg\min_h \mathbb{E}[\mathcal{L}(h_\Theta(\mathbf{X}), \mathbf{Y})] \tag{1}$$

using stochastic gradient descent

$$\Theta_{t+1} = \Theta_t - \eta \nabla \mathcal{L}(h_\Theta(\mathbf{X}), \mathbf{Y}) \tag{2}$$

Now, such a setup only considers local information. That is, the loss is computed on a per-image basis. Typically, such a loss function is used as a surrogate for optimizing for other measures that can be called ”global” across the dataset, such as the precision-recall curve, the receiver operating characteristic, or the F_1 score. This surrogate may not be very good, and might lead to inferior results [2] [3].

In this paper we explore the following question: what if there are certain signals that are only available on the global level? If so, is there a way for us to feed this information into the learning process? We set our scopes on the issue of using such a global signal as a regularizer.

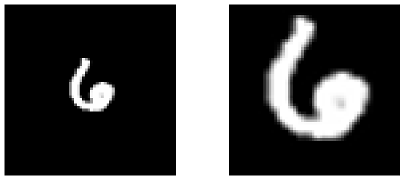
(a) A Small ”6” (b) A Big ”6”

Figure 1: We double the size of the MNIST [6] dataset by making ”small” and ”big” versions of each image. A classifier that is trained on such a dataset should ideally treat both the ”small” and ”big” versions of each image in the ”same manner”. However, in practice this will not happen unless we ”normalize” it somehow using a ”global” signal because such intuition of smallness or bigness is not fed into the learning process on a per-image or per-minibatch basis.

To provide some inspiration for this question, we begin by considering the elementary example in 1. A more practical example of semantic segmentation is described in 2. Furthermore, we describe how this framework might also be useful in a ”Positive-Unlabeled” (PU) setting in 3. Finally, we introduce an interesting and important area of application for F1 regularization in ML fairness. We do believe that we are only touching the tip of the iceberg with respect to applications for our proposed procedure [4][5].

1.2. Hypothesis

We hypothesize that if our loss function were also able to incorporate global information, i.e. information that is found only across the dataset but not on a per-image basis, then we would be able to better solve such a problem.

Hence, we propose having the objective

$$\mathcal{L}(h_\Theta(\mathbf{X}), \mathbf{Y}) + \beta F_1 \tag{3}$$

where F_1 is the F_1 classification score of a “large” vs. “small” image. We now have a certain multi-task formulation where we incorporate both local and global information in our loss function.

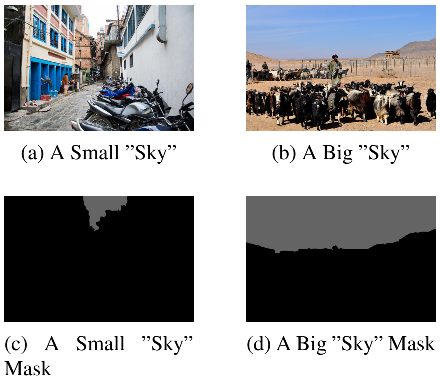
(a) A Small "Sky" (b) A Big "Sky" (c) A Small "Sky" Mask (d) A Big "Sky" Mask

Figure 2: We take a look at some examples from the MS COCO dataset [7]. Here, we have the same class: sky, but in different sizes. We want to ensure that our classifier treats both these objects as the same, but without some sort of "global" normalization, it would not be possible to feed such information into the learning process.

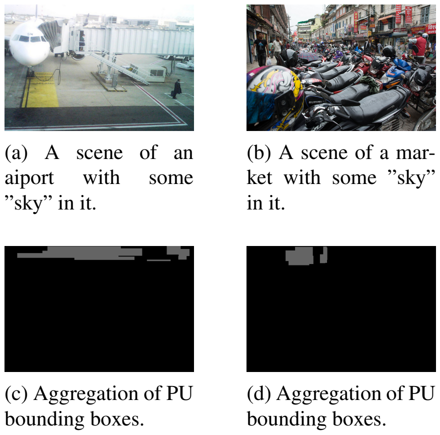
(a) A scene of an aiport with some "sky" in it. (b) A scene of a market with some "sky" in it. (c) Aggregation of PU bounding boxes. (d) Aggregation of PU bounding boxes.

Figure 3: In this setting, we have a very weak type of supervision. We consider only the scene points inside the bounding boxes to be positive; everything else is unlabeled. Using a global information measure, we can sidestep the problem of having to learn from a single image. If we had to learn from a single image or mini-batch, the model might get confused between sky and not-sky. But on a global level, it is still clear what pixels must be classified as sky.

This is not a trivial task because the F_1 score is not trainable by stochastic gradient descent. Hence, we transform the F_1 score into something that will be amenable to training by stochastic gradient descent (and other optimizer such as Adam).

Figure 4: Consider the ML fairness context. We might have certain attributes such as gender, age, or race that we might want to "normalize" or "regularize" against. In other words, we wish to tell our classifier to learn that it should teach each person from each protected attribute (e.g. race) in the "same manner". (Image from [8])

2. Approach

2.1. Defining the F1 Score

There is the standard definition for the F_1 score:

$$F_1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

but this is not a decomposable objective that can be scalably optimized [1]. Hence, we instead opt for optimizing a lower bound to this F_1 score built using the hinge loss. We omit derivation details for the linear program as this information is available in [1].

Our contributions begin with considering the linear program as given in [1] in the section on the F_β score solved for $\beta = 1$ [1]. We replace Y with I in the given linear program. Then, solving the following linear program is equivalent to finding a tightest lower bound for the F_1 score:

$$\min_{\phi,\tau,w,\epsilon} |I^+|\epsilon + \sum_{i \in I^+} \tau_i + \sum_{i \in I^-} \phi_i \tag{5}$$

$$\text{s.t. } \forall\ i \in I^+, \tau_i \leq \epsilon, \tau_i \leq w \cdot x_i \tag{6}$$

$$\forall\, i \in I^-, \phi_i \geq 0, \phi_i \geq \epsilon + w \cdot x_i \tag{7}$$

$$\sum_{i \in I^+} \tau_i = 1 \tag{8}$$

$$\epsilon \geq 0. \tag{9}$$

2.2. Multi-Task Architecture

Let $h_{\theta_1,\ldots,\theta_{n-1}}$ denote a deep neural network with $n-1$ layers. The nth layer is typically a layer that prepares the features for the final task like classification (e.g. a fully connected layer with C channels for C classes). We utilize this

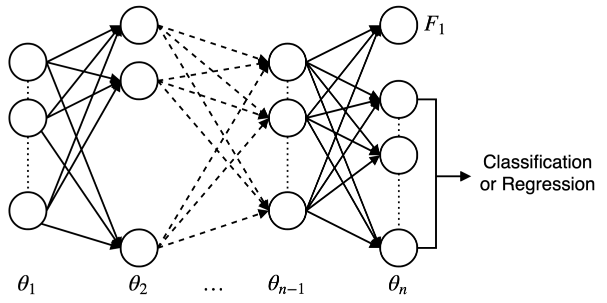

Figure 5: We consider any arbitrary neural network with layer weights $\theta_1, \theta_2, \ldots, \theta_{n-1}$. We propose a multi-task setting wherein the last layer is used for both the primary task (such as regression or classification), and the F_1 regularization. All model weights are trained simultaneously for both tasks.

notation because we have a multi-task objective: depending on the final output layer, we can either perform regression, classification, or even segmentation alongside the F_1 part 3. Hence, let $fc^{(k)}_{\theta_n}$ denote a fully connected layer with k outputs parametrized by the nth layer weights θ_n. For brevity, we can then denote $h^{(F_1)}_{\Theta}$ as the model output for the F_1 task, where $h^{(F_1)}_{\Theta} \triangleq fc^{(1)}_{\theta_n}(h_{\Theta_{1:n-1}})$.

In the above linear program, w acts as a type of separating hyperplane as in the hinge loss. Accordingly, x_i acts as the input feature to the classifier, and thus, we can replace x_i by $h^{(F_1)}_{\Theta}(\mathbf{X}_i)$. This new formulation allows for feature learning before classification by w. Hence, we have the following linear program:

$$\min_{\phi,\tau,w,\epsilon} |I^+|\epsilon + \sum_{i\in I^+} \tau_i + \sum_{i\in I^-} \phi_i \tag{10}$$

$$\text{s.t. } \forall\ i \in I^+, \tau_i \leq \epsilon, \tau_i \leq w \cdot h^{(F_1)}_{\Theta}(\mathbf{X}_i) \tag{11}$$

$$\forall\, i \in I^-, \phi_i \geq 0, \phi_i \geq \epsilon + w \cdot h^{(F_1)}_{\Theta}(\mathbf{X}_i) \tag{12}$$

$$\sum_{i\in I^+} \tau_i = 1 \tag{13}$$

$$\epsilon \geq 0. \tag{14}$$

2.3. Solving the F1 score lower bound

Firstly, notice that we can eliminate the $\sum_{i\in I^+} \tau_i$ term from the objective function (10) since any feasible solution has to satisfy the constraints in (13). Secondly, using Hinge loss, we can eliminate ϕ_i terms in the objective function too, specifically by replacing ϕ_i with $\max(0, \epsilon + w \cdot h^{(F_1)}_{\Theta}(\mathbf{X}_i))$. Hence we can rewrite the LP as:

$$\min_{\tau,w,\epsilon,\Theta} n\epsilon + \sum_{i\in I^-} \max(0, \epsilon + w \cdot h^{(F_1)}_{\Theta}(\mathbf{X}_i)) \tag{15}$$

$$\text{s.t. } \forall\, i \in I^+, \tau_i \leq \epsilon, \tau_i \leq w \cdot h^{(F_1)}_{\Theta}(\mathbf{X}_i) \tag{16}$$

$$\sum_{i\in I^+} \tau_i = 1 \tag{17}$$

$$\epsilon \geq 0. \tag{18}$$

In order to subject this optimization problem to stochastic gradient descent, we dualize. Hence, we get:

$$\max_{\lambda\geq 0,\mu,\gamma} \min_{\tau,\epsilon,w,\Theta} L(\tau, w, \Theta, \epsilon, \lambda, \mu, \gamma) \tag{19}$$

where

$$\begin{aligned} L(\tau, w, \Theta, \epsilon, \lambda, \mu, \gamma) = n\epsilon &+ \sum_{i\in I^-} \max(0, \epsilon + w \cdot h^{(F_1)}_{\Theta}(\mathbf{X}_i)) \\ &+ \mu((\sum_{i\in I^+} \tau_i) - 1) \\ &+ \sum_{i\in I^+} \lambda_i(\tau_i - w \cdot h^{(F_1)}_{\Theta}(\mathbf{X}_i)) \\ &+ \sum_{i\in I^+} \gamma_i(\tau_i - \epsilon) \end{aligned} \tag{20}$$

Then, we run vanilla stochastic gradient descent on the primal variables $\tau, \epsilon, w, \Theta$, and we project $\epsilon = \max(0, \epsilon)$ to ensure non-negativity.

Finally, we run dual updates:

$$\mu = \mu + \eta_\mu \left(\left(\sum_{i\in I^+} \tau_i \right) - 1 \right)$$

$$\lambda_i = \lambda_i + \eta_\lambda \left(\tau_i - w \cdot h^{(F_1)}_{\Theta}(\mathbf{X}_i) \right)$$

$$\gamma_i = \gamma_i + \eta_\gamma(\tau_i - \epsilon)$$

The summations run over minimatches while running stochastic gradient descent. The algorithm can be summarized as in algorithm 1:

3. Evaluation

3.1. Implementation and Training Details

We implement our algorithm in PyTorch [9]. We run our algorithm for 40 outer loops, and 600 inner loops for following experiments. We use the built-in stochastic gradient descent optimizer of PyTorch with learning rate 0.01. For our hyperparameters, we use $\beta = 0.001, \eta_\tau = 0.01, \eta_\epsilon = 0.01, \eta_w = 0.01, \eta_\gamma = 0.001, \eta_\lambda = 0.001, and \eta_\mu = 0.00001$.

```
for $i = 1, \ldots, N_{outer}$ do
    for $i = 1, \ldots, N_{inner}$ do
        Sample a minibatch;
        Run the minibatch through the model;
        Compute the multitask loss 3;
        Perform primal updates;
        $\epsilon = \max(0, \epsilon)$;
    end
    Perform dual updates;
end
```

Algorithm 1: Dual Ascent Procedure

We also experiment with various schedules on the number of inner loops due to the dual ascent procedure outlined above. The intuition is that when the dual parameters have not yet been set in, the inner loops produce bad approximations to the true primal variables. Hence, it is optimal to have fewer inner loops initially. Once the dual variables have been learned, it is optimal to get a better primal approximation, and hence run for a larger number of inner loops. We have not been able to find any serious improvement through this process yet so we do not report any results on this matter.

3.2. MNIST

We begin by evaluating our algorithm on the MNIST dataset. We construct our dataset following the example 1 in our introduction section. To do so, we take each image in the MNIST dataset, and create two versions of it: a "big" one, and a "small" one. We utilize a single fully connected layer neural network for evaluation. Our choice is motivated by the fact that for most deeper models we achieved a 98% or more accuracy, and it becomes unclear how we can evaluate our algorithm. We present results in 6.

3.3. Stuff Segmentation

We define "stuff" as objects like "sky", "grass", or "concrete", as opposed to "things" like "car", "tree" or "person". For our evaluation, we focus only on a foreground-background segmentation task selecting the "sky" as an example (as in 2) for simplicity of preparing the dataset (code publicly available at [10]). Our method generalizes easily to multi-class segmentation. To define "big" and "small", we draw a histogram 7 and follow the procedure outlined in the description of the histogram. We utilize SegNet [11], a deep encoder-decoder model, as our model for this task. We again find that F_1 normalization is useful for stuff segmentation, and that our algorithm outperforms the baseline.

4. Conclusion

We have demonstrated that F_1 regularization can be both useful and important in the domains of image classification

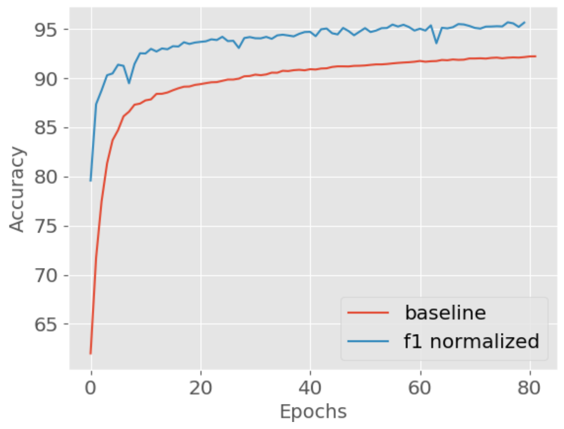

Figure 6: The F_1 normalization approach vastly outperforms the vanilla baseline that only considers per-image information.

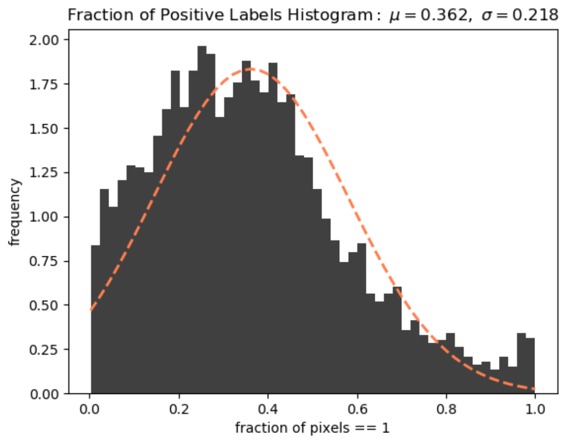

Figure 7: We define our "big" and "small" classes for our stuff segmentation task by observing this histogram. We set all those images with "fraction of pixels == 1" ≥ 0.6 as "big", and all the other images as "small".

and segmentation. However, we believe that we have only touched the tip of the iceberg. We imagine there to be many more applications that we have not yet come up with. For example, F_1 score optimization has been used in diverse applications such as adversarial prediction games [4]), and in automatic mispronunciation detection [5].

Our algorithm allows scalable optimization of the F_1 score alongside any primary task such as classification or regression. We plan to further explore more theoretical aspects of our algorithm, such as convergence tests and performance analysis. We will perform a more systematic hyperparameter search using techniques such as HyperBand

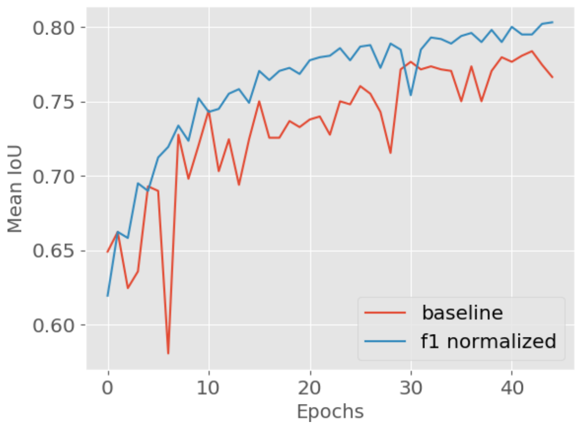

Figure 8: We find that F_1 normalization improves the mean IoU of output masks by 4%.

[12]. Finally, we our most excited about possible utility in the domain of ML fairness as we believe that such an F_1 regularization method can provide both important theoretical guarantees for fairness, as well as powerful applications that are highly in demand today in industry [4].

5. Acknowledgements

The content in this project is inspired by research conducted in Dr. Vikas Singh's (`vsingh.cs.wisc.edu`) group with Sathya N. Ravi (`ravi5@wisc.edu`). Content from this project will be used towards a conference publication.

References

[1] E. E. Eban, M. Schain, A. Mackey, A. Gordon, R. A. Saurous, and G. Elidan, "Scalable learning of non-decomposable objectives," *arXiv preprint arXiv:1608.04802*, 2016.

[2] C. Cortes and M. Mohri, "Auc optimization vs. error rate minimization," in *Advances in neural information processing systems*, 2004, pp. 313–320.

[3] Y. Yue, T. Finley, F. Radlinski, and T. Joachims, "A support vector method for optimizing average precision," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2007, pp. 271–278.

[4] J. Dastin, "Amazon scraps secret ai recruiting tool that showed bias against women," Oct 2018.

[5] H. Huang, H. Xu, X. Wang, and W. Silamu, "Maximum f1-score discriminative training criterion for automatic mispronunciation detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 4, pp. 787–797, 2015.

[6] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *AT&T Labs*, vol. 2, p. 18, 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist

[7] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[8] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," *arXiv preprint arXiv:1812.04948*, 2018.

[9] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[10] A. Venkatesh, "Coco stuff tools," 2019. [Online]. Available: https://github.com/abhay-venkatesh/coco-stuff-tools

[11] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[12] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *arXiv preprint arXiv:1603.06560*, 2016.